Tree-Shellable論理関数の判定の複雑さ (計算理論とアルゴリズムの新展開)

AUTHOR(S):

門野, 伸史; 武永, 康彦

CITATION:

門野, 伸史 ...[et al]. Tree-Shellable論理関数の判定の複雑さ (計算理論とアルゴリズムの新展開). 数理解析研究所講究録 2001, 1205: 83-88

ISSUE DATE:

2001-05

URL:

http://hdl.handle.net/2433/41005

数理解析研究所講究録 1205 巻 2001 年 83-88

Tree-Shellable論理関数の判定の複雑さ

門野 伸史 (Shinji Kadono)　武永康彦 (Yasuhiko Takenaga)
電気通信大学大学院電気通信学研究科情報工学専攻

概要

Tree-shellable 論理関数とは、ある関数 f の主項の数と、その関数を表している二分木に特別な関係を持っている関数である。その特長として、関数 f の双対関数が容易に求めることができる。また、信頼性の計算にも有効である。しかし特長を生かすには、Tree-shellable 論理関数であるか否かの判定が必要である。現在、与えられた関数が tree-shellable であるか否かの判定は、NP に含まれていることは明らかだが、NP 完全であるかどうかは不明である。各積項のリテラル数が 2 以下の時は $O(mn^2)$ での判定が可能である。また項順序と変数順序が決まっていれば、その順序の下で tree-shellable であるか否かの判定は多項式時間で可能である。本研究では、各積項のリテラル数が 2 から成る関数に 3 つのリテラルから成る項を 1 つ加えた関数が tree-shellable 論理関数であるか否かを判定するアルゴリズムを組み立て、更にその判定時間が $O(mn^3)$ で可能であることを証明した。

1　はじめに

論理関数の特性を理解し、明らかにすることは、計算機科学の分野において重要なことである。積和形論理式 (Disjunctive Normal Form Boolean formula:DNF) は、論理関数の表現法である各積項は、論理関数の内項に相当し、特に主項は、論理関数を最小積和形で表わすなど非常に重要な概念である。

Tree-shellable 論理関数とは、その関数の主項と、その関数を表わす二分木の根から 1 ノードへのパスの数が等しいような二分木を持つ関数である。Ordered tree-shellable 論理関数は、tree-shellable 論理関数の一種で、その関数を表わした二分木が、順序付二分木であるという制約を持つものである。

関数が二分木で表わされていれば、第一に、関数 f の積和形表現から双対関数 $f^d = \overline{f(\overline{x_1}, \cdots, \overline{x_n})}$ の積和形表現を容易に求めることができる。二分木の 0 枝と 1 枝を逆にして 0 ノードと 1 ノードを交換するだけで、f^d の二分木表現が得られる。関数 f が、tree-shellable であれば f の二分木表現も容易に求まることから多項式時間で、f^d の DNF 表現を求めることができる。

第二に、ある種のシステムの信頼性を計算する Union of Product 問題 [1] を容易に解くことができる。入出力は以下の通りである。

入力 $Pr[x_i = 1](1 \leq i \leq n), f(x_1, \cdots, x_n)$

出力 $Pr[f(x_1, \cdots, x_n)]$

Pr は確率を表わし、それぞれの変数は、各サブシステムが動作しているか否かを表わし、これから全体の信頼性を求める問題と考える。もし f が tree-shellable ならば、othogonal に変形できることを利用して正確な $Pr[f = 1]$ の値を容易に計算できる。(DNF が orthogonal であるとは、どのような割当に対しても高々1 つの項が値 1 を持つことである。) このような tree-shellable 論理関数の特長を生かすには関数が tree-shellable であるか否かの判定が必要である。現在、与えられた関数が ordered tree-shellable であるか否かの判定は、各積項のリテラル数が 5 以上の時は NP 完全であることが示されている [3]。また、各積項のリテラル数が 2 以下の時は多項式時間での判定が可能である [2]。また、変数順序が決まっていればその順序の下で ordered tree-shellable であるか否かの判定は多項式時間で可能である。与えられた関数が tree-shellable であるか否かの判定は、NP に含まれていることは明らかだが、NP 完全であるかどうかは不明である。各積項のリテラル数が 2 以下の時は多項式時間での判定が可能である。変数順序と項順序が決まっていればその順序の下で多項式時間での判定が可能である。本研究では、quadratic 論理関数に、3 つのリテラルから成る項を 1 つ加えた関数の tree-shellable 判定の複雑さを明らかにする。

2 準備

2.1 基本表記

論理関数における基本的な定義を行う。$B=\{0,1\}$、nは自然数の集合、$[n]=\{1,2,\cdots,n\}$ただし$[0]=\emptyset$とする。n変数論理関数とは、写像$f:B^n\to B$である。$x_1,x_2,\cdots,x_n$をfの変数とし、$\overline{x_i}(i\in[n])$はx_iの否定とする。

πは$[n]$上の置換、$\pi(i)$はπのi番目の整数とする。πの中でsがtより先にある場合は、$s\prec_\pi t$とし、$\min_\pi(S)$と$\max_\pi(S)$を次のように定義する。

- $h\in S$であり、すべての$i\in S\backslash h$に対し$h\prec_\pi i$である時、$\min_\pi(S)=h$。
- $h\in S$であり、すべての$i\in S\backslash h$に対し$i\prec_\pi h$である時、$\max_\pi(s)=h$。

I_s,I_tを$[n]$の互いに異なる部分集合とし、$\pi(i)$を$[n]$の順列とする。ある$i\in[n]$に対し、$I_s\cap\{\pi(1),\cdots,\pi(i-1)\}$ $I_t\cap\{\pi(1),\cdots,\pi(i-1)\}$、$\pi(i)\in I_s$、$\pi(i)\notin I_t$、$I_s\prec_L I_t$とする。これを$\pi$についての辞書式順序という。また、$I$の要素数を$|I|$で表わす。

2.2 積和形論理式

$f(x_1,\cdots,x_n)$を論理関数とする。$g(\mathbf{x})=1$となる割当て$\mathbf{x}\in\{0,1\}^n$が全て$f(\mathbf{x})=1$を満たす時$f\geq g$とする。$I\subseteq[n]$とする時、$\bigvee_{i\in I}x_i\leq f$を満たす$\bigvee_{i\in I}x_i$を$f$の内項という。また、内項が任意の$j\in I$に対して$\bigvee_{i\in I-\{j\}}\not\leq f$を満たす時、$f$の主項という。

$f=\bigvee_{k=1}^{m}(\bigwedge_{i\in I_k}x_i\bigwedge_{j\in J_k}\overline{x_j})$の形で表わされた式を積和形論理式 (Disjunctive Normal Form Boolean formula : DNF) と呼ぶ。ただし、$k=1,\cdots,m$に対し$I_k,J_k\subseteq[n]$、$I_k\cap J_k=\emptyset$とする。また、$T_k=\bigwedge_{i\in I_k}x_i\bigwedge_{j\in J_k}\overline{x_j}$を$f$の項という。

全ての項の対$T_k,T_l(k\neq l)$が、$(I_k\cap J_l)\cup(I_l\cap J_k)\neq\emptyset$を満たすDNFをOthogonal DNF(ODNF)という。fがODNFであるならば、どのような割当てに対しても高々1つの項が値1を持つ。$I_k\subseteq[n]$とする時$f=\bigvee_{k=1}^{m}\bigwedge_{i\in I_k}x_i$の形を積和形正論理式 (Positive DNF : PDNF) と呼ぶ。

いくつかの項の対$T_k,T_l(k\neq l)$が、$I_k\subseteq I_l$をみたすPDNFを冗長なDNF(redundant PDNF)という。またどんな項の対$T_k,T_l(k\neq l)$も$I_k\subseteq I_l$を満たさないPDNFを非冗長なPDNF(irredudant PDNF)という。以後、本論文で現れる全ての論理関数はは非冗長なPDNFであるものとする。

2.3 二分決定木

二分決定木 (Binary Deciesion Tree : BDT) は論理関数を表わす、ラベル付けされた木である。BDTの葉ノードには0か1によってラベル付けされており定数ノードと呼ぶ。他のノードは変数がラベル付けされおり、変数ノードと呼ぶ。$label=(v)$をノードvのラベルとする。葉ノード以外のノードからは2本の枝が出ており、0枝、1枝と呼ばれる。$edge_0(v),edge_1(v)$をそれぞれノードvから0枝、1枝を指しているノードと表記する。関数の定数は、根ノードから葉ノードへたどることにより与えられる。ノードにおいて、2つの枝の1つが変数の割当てに従って選ばれる。葉ノードのラベルが0ならば、関数の定数は0で、葉ノードのラベルが1ならば、関数の定数は1である。

根ノードから1とラベル付けされた葉ノードまでのパスを1パスという。1パスPはリテラルの列で表される。Pのk番目の枝がラベルx_iであるノードから出た1枝 (0枝) ならば正のリテラルx_i(負のリテラル$\overline{x_i}$)

が P の k 番目の要素である。$\tilde{x_i}$ が P を表わす列に属する時、$\tilde{x_i} \in P$ と表示する。$\tilde{x_i}$ は、x_i または、$\overline{x_i}$ のいずれかを指す。どのパスにおいても、各変数は高々1回しか現れない。

ノード v から出た0枝、1枝が同じ関数を表わすノードを指している時、v は冗長ノードといい、冗長ノードを持たない BDT を rededced BDT という。この論文では、BDT は redeced BDT を意味している。

BDT の全てのパスにおいて、現れる変数の順序が π に矛盾なく従っている場合、その BDT を Ordered BDT(OBDT) という。π を OBDT の変数順序とする。

f_v を v で表わされた論理関数とすると f_v は次のように定義される

$f_v = label(v) \cdots v$ が定数ノードの時

$f_v = \overline{label(v)} \cdot f_{edge_0(v)} + label(v) \cdot f_{edge_1(v)}$

$\cdots v$ が変数ノードの時

BDT T は、k 個の1パス $P_1, \cdots, P_k$ を持つものとし、$I_k = \{i \mid x_i \in P_k\}, J_k = \{i \mid \overline{x_i} \in P_k\}$ とする。この時、T を表わす関数は、$f = \bigvee_{k=1}^{m} (\bigwedge_{i \in I_k} x_i \bigwedge_{i \in J_k} \overline{x_j})$ で表わされる。

3 Tree-Shellable 論理関数

3.1 Tree-Shellable 論理関数

定義 1. [4] f は正論理関数で PDNF

$$f = \bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$$

で表わされているとする。f を表わす BDT T で、m 個の1パス $P_1, \cdots, P_m$ を持つものが存在する時、f は tree-shellable である。

どの1つの主項に対しても $i \in I_k \Leftrightarrow x_i \in P_k$ の関係を持つ1パス P_k が1つ存在する。この P_k を項 I_k に対応する1パスと呼ぶ。任意の異なる1パスの対 P_s, P_t に対して P_s, P_t は、根ノードから出発してノード v までは同じルートをたどるが、ノード v から P_s は1枝を通り、P_t は0枝を通るパスとなるような x_i でラベル付けされたノード v が存在する。これは、$I_s \wedge J_t \neq \emptyset$ を導く。すなわち $f = \bigvee_{k=1}^{m} (\bigwedge_{i \in I'_k} x_i \bigwedge_{j \in J_k} \overline{x_j})$ は、ODNF である。これは関数を二分木で表わしていることからも明らかである。また、この2つの1パスの関係は、P_t は P_s より左側に現れることから、P_t は P_s の左側にあると表記する。また1パスが BDT の1番左に現れることは、BDT で表わした時これ以上左には1パスが BDT には存在しないことをいう。

定理 1. [4] $f = \bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$ が tree-shellable であるとする。この時、BDT T が f が tree-shellable であることを示す必要十分条件は、全ての P_k および、$\overline{x_i} \in P_k$ を満たす全ての i に対し、$I_l \subset I_k \cup \{i\}$ を満たす I_l が存在することである。

定理1を利用することにより、与えられた PDNF が tree-shellable であるか否か判定できる。最初に根ノードのラベル x_i を与え、定理1の条件を満たしているか判定する。条件を満たせば、2つの部分関数 $f|_{x_i=0}, f|_{x_i=1}$ が共に tree-shellable であるか否か判定する。共に tree-shellable ならば、f は tree-shellable であり、且つ x_i とラベル付けされた根ノードを持つ1パスの条件を満たした BDT も存在する。

3.2 Ordered Tree-Shellable 論理関数

定義 2. [4] f は正論理関数で、PDNF

$$f = \bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$$

で表わされているものとする。f を表わす変数順序が π である OBDT T で m 個の 1 パス $P_1, \cdots, P_m$ を持つものが存在する時 f は π に関して ordered tree-shellable である。またこのようなが π が存在する時、f は ordered tree-shellable である。

定理 2. [4] 以下の条件を満たす時、その時に限り論理関数 f は、π に関して ordered tree-shellable である。$\pi(i) \notin I_k, \pi(i) \prec_\pi \max(I_k)$ を満たす全ての項 T_k と i に対して 1 または 2 のどちらかが成り立つ。

1. $(I_k \cup \{\pi(i)\}) \cap \{\pi(1), \cdots, \pi(i)\} = I_l \cap \{\pi(1), \cdots, \pi(i)\}$ を満たす I_l が存在しない。
2. このような I_l が存在するならば、少なくとも 1 つは $I_k \cup \{\pi(i)\} \supset I_l$ を満たす。

4 判定の複雑さ

4.1 Quadratic 論理関数

各積項が 2 つのリテラルから成る論理関数、つまり、どのような $k \in [m]$ に対しても $|I_k| = 2$ を満たす積和形論理式 $\bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$ で表わされる関数を quadratic 論理関数という。Quadretic 論理関数は、グラフ $G = (V, E)$ として表わすことができる。ノードは変数、枝は項にそれぞれ相当し、$V = [n], I_k = \{i, j\}$ となる k が存在する時、$(i, j) \in E$ とする。$V' \subseteq V$ により導かれる G のサブグラフ $G' = (V', E')$ は次のように定義される。

$$V' \subseteq V, E' = \{(i, j) | i, j \in V', (i, j) \in E\}$$

また、V' により導かれるグラフが長さ $|V'|$ のサイクルを作る時、これを induced サイクルという。

長さが 4 以上の induced サイクルを含まないグラフを triangulated グラフという。Triangurated グラフの補グラフを cotriangulated グラフといい、全ての induced サブグラフが cosimplicial ノードを含む。Cosimplicial ノードとは、そのノードと隣接していないノードの集合による induced サブグラフが、独立点集合となるノードである。

定理 3. [2] Quadratic 論理関数 $f_q = \bigvee_{(i,j) \in E} x_i x_j$ は、$G = ([n], E)$ が、cotriangulated グラフである時、その時に限り、ordered tree-shellable である。

Quadratic 論理関数が ordered tree-shellable の場合、その変数順序は次のように求めることができる。

1. Cosimplicial ノードまたは、枝が出ていない独立したノードを選ぶ。
2. 選んだノードと、そのノードから出る枝を削除する。
3. 全てのノードが選ばれるまで、1,2 を繰り返す。

定理 4. [4] Tree-shellable quadratic 論理関数は、ordered tree-shellable である。

系 1. *Quadratic* 論理関数に対して、f_q が *ordered tree-shellable* であるか否かの判定とその変数順序を求めるには、$O(mn^2)$ 時間で判定が可能である。更に *tree-shellable* であるか否かの判定も定理 4 より、$O(mn^2)$ 時間で可能である。

4.2 判定アルゴリズム

そこで、quadratic 論理関数に、3 つのリテラルから成る項 K を 1 つ加えた関数 f に対する tree-shellable であるか否かの判定問題を考える。つまり、このような形をした関数 f は、$f = f_q + K$ と表わせる。Tree-shellable であるか否かの判定アルゴリズムは以下のようになる。なお K に対応した 1 パスを P_K とする。

1. Quadratic 論理関数 f_q が tree-shellable であるか否かを判定する。f_q が tree-shellable でなければ、f は tree-shellable でない。

2. f_q と K の共通変数のノードの集合を A とし、ノード $x_a(\in A)$ と隣接しているノードの集合を B とする。$A=\emptyset$ ならば、f は tree-shellable でない。
3. 集合 A の中で、cosimplicial ノードとなるノード x_a が存在すれば f は tree-shellable である。そうでなければステップ 4 へ。
4. 集合 B 中で、cosimplicial ノードとなるノード x_b があれば、x_b 及び x_b と隣接している枝を消去する。もし cosimplicial なノードが複数存在すれば、まとめて全て消すことができる。そしてステップ 3 へ戻る。集合 B の中に cosimplicial ノードが無いならば、f は tree-shellable でない。

アルゴリズムの流れを説明する。ステップ 1 では、quadratic 論理関数 f_q をグラフ G で表わし、cotriangulated グラフであれば、ステップ 2 へ移る。集合 A,B を見つけてステップ 3 へ移り、ステップ 3,4 では cosimplicial ノードとなる変数を A,B から探す。ステップ 3,4 を繰り返し、最終的には、cosimplicial ノードが集合 A の中に存在すれば f は tree-shellable である。

以下では、上のアルゴリズムにより f が tree-shellable であるか否か正しく判定できることを証明する。これは、以下の定理 5,6,7 により示される。

定理 5. Quadratic 論理関数 f_q が tree-shellable でなければ、関数 f も tree-shellable でない。

証明. まず、関数 f_q が BDT T を表わしているものとする。Quadratic 論理関数 f_q が、tree-shellable でないと仮定すると、f_q の任意の積項 I_h に対し、I_h に対応した 1 パスを $P_h(1\leq h\leq m)$ とする。$P_{k_1},\cdots,P_{k_t}$ は、x_i によりラベル付けされたノードから 1 枝を通る 1 パスとすると、ある I_h と i に対し定理 1 の条件 $I_{k_p}(1\leq p\leq t)\subset I_h\cup\{i\}$ を満たさない。項 K に対応する 1 パス P_K を加えて定理 1 の条件を満たすには、$K=I_h\cup\{i\}$ となる必要がある。しかし、この時 $K\supset I_h$ を満たすので冗長な論理関数となる。ゆえに、関数 f は tree-shellable でない。 □

定理 6. f_q と K との間に共通変数が存在しないならば、f は tree-shellable でない。

証明. 関数 f が BDT T を表わしているものとする。2 つの場合に分けて証明をする。

1.$K=x_\alpha x_\beta x_\gamma$ の 1 パス P_K が根ノード以外に現れる時

P_l を P_K より右の 1 パスとする。この時あるノード v に対し、P_l と P_K は v で分岐し、P_l は v から 1 枝を通り、P_K は v から 0 枝を通る。この時 I_l と K との間に共通変数が存在しないので、$I_l\subset K\cup\{i\}$ を満たすことはない。ゆえに、f は tree-shellable でない。

2.$K=x_\alpha x_\beta x_\gamma$ のいずれかを根ノードとした時

I_l を f_q の任意の項 I_l に対応する 1 パスを P_l とする。一般性を失うことなく x_α を根ノードのラベルと仮定する。この時、f_q の任意の項 I_l は、対応するパスが $\overline{x_\alpha}$ を含むことから、$I\subseteq I_l\cup\{\alpha\}$ を満たす項 I が存在することが必要である。しかし、x_α を含む項は K しかなく、また f_q の項と K との間に共通変数が存在しないことから、これを満たす項 I は存在しない。f_q の任意の項 I_l は、α を含まない。ゆえに、f は tree-shellable でない。 □

次の定理に入る前に、新たに定義を与える。

定義 3. 関数 f が BDT T により表わされているとする。K に対応した 1 パス P_K があって、P_K が初めて 1 枝を通るノードを v とすると、それより上のノードをストリートとする。

定理 7. BDT T が、f が tree-shellable であることの必要十分条件は、BDT T のストリートの任意の変数が、残りのグラフについて cosimplicial かつ集合 B に属していることである。

証明. 関数 f を BDT T で表わしているものとする。

(必要条件)

2 つの場合に分けて証明をする。ストリート上の x_s によりラベル付けされたノードが、

1. 集合 B に属していない時

集合 A,B に含まれていないノードの集合を C とすると、頂点 $x_c \in C$ は、集合 A と隣接していない。つまり、x_c を含んだ項と K との間には共通のノードが存在しない。更に、x_c を含んだ任意の項 I_c は、定理 1 の条件 $I_c \subset K \cup \{c\}$ を満たすことができない。ゆえに、関数 f は tree-shellable でない。

2. 集合 B に属しているが、cosimplicial ノードでない時

cosimplicial ノードでない $x_b \in B$ によってラベル付けされたノードがストリート上に現れた時、x_b を含んだ項と K との間には共通のノードが存在し、x_b を含んだ任意の項 I_b が、定理 1 の条件 $I_b \subset K \cup \{b\}$ を満たす。しかし、K に対応する 1 パス P_K を省くと quadratic 関数 f_q の二分木表現と同じになる。変数 x_b が cosimplicial ノードでないならば、P_K を除いてできる BDT も f_q が tree-shellable であることを示さない。更に、定理 5 を利用すると f も tree-shellable でないことがいえる。

(十分条件)

ストリート上のノードは cosimplicial ノードであり、集合 B に属する変数によりラベル付けされているとする。変数 $x_b \in B$ を含んだ任意の項 I_b と、K との間には共通変数が存在するので、K に対応した 1 パスが存在し、$I_b \subset K \cup \{b\}$ を満たす。最後に、集合 A に含まれているノードが cosimplicial ノードとなれば、K の 1 パス P_d は x_a から 1 枝を通りその部分木の一番左に表せばよい。ゆえに関数 f は tree-shellable である。 □

このアルゴリズムの判定時間は、ステップ 1 は $O(mn^2)$ 時間、ステップ 2 では $O(m)$ 時間、ステップ 3,4 では $O(n)$ かかる。更に新たに cosimplicial ノードを求め直すのに $O(mn^3)$ 時間かかるので、合計で $O(mn^3)$ 時間の判定が可能となる。

5 考察と今後の課題

今回の研究では、quadratic 関数 f_q に、3 つのリテラルから成る項 J_1 を 1 つ加えた関数 f は、多項式時間で判定が可能であることがわかった。今後の課題としては、quadratic 関数に 3 つのリテラルから成る項を 1 つ加えた関数の ordered tree-shellable 論理関数の判定時間、そして 2 つ以上加えた関数の tree-shellable 論理関数及び ordered tree-shellable 論理関数の判定時間、更に各積項のリテラル数が 3,4 の時の ordered tree-shellable 論理関数の判定時間を考えることである。

参考文献

[1] M.O.Ball and J.S.Provan,"Disjoint Products and Efficient Computation of Reliability",Operations Reserch,Vol36,No5,pp703-715(1988).

[2] C.Benzenkin,Y.Crama,P.Duchet,P.L.Hammar and F.Maffray,"More Characterization of Triangulated Graphs",Journal of Graph Theory,Vol.14,N0.4,pp.413-422

[3] E.Boros,Y.Crama,O.Ekin,P.L.Hammer,T.Ibaraki and A.Kogan,"Boolean Nomal Forms, Shellability and Reliability Computations",SIAM J.Discrate Mathematics,Vol.13,pp.212-226(2000).

[4] Y.Takenaga,K.Nakajima and S.Yajima,"Tree-Shellability of Boolean Functions" Theoret.Comput.Sci.(to appear) .

[5] 宮山一幸,"Shellable 論理関数の認識の複雑さ"電気通信大学卒業論文 (1998).